CLAIMS

What is claimed is:

1. A method for handling events in a data processing system in which at least two threads access the same data, said method comprising the step of handing off the task of accessing said data from a first event handler to a second event handler that is already accessing said data.

2. The method of claim 1 in which a compare and swap function is used to insure that data that is written between threads is not lost when more than one thread attempts to write to a variable at the same time.

3. The method of claim 2 in which a bit array is updated by the compare and swap function so as to contain a list of events to be handled by the thread that is currently handling events.

4. The method of claim 3 in which the bit array includes an indicator which signifies that no events are waiting to be handled and that no thread is currently handling events.

5. The method of claim 3 in which the bit array includes an indicator which signifies that no events are waiting to be handled and that a thread is currently handling events.

6. The method of claim 3 in which the compare and swap instruction is employed to change the bit array from the "no events and no thread handling events" indication to the "no events and a thread is handling events" indication.

7. The method of claim 6 in which the bit array is changed to the "no events and a thread is handling events" indication so that the thread that makes that change handles subsequent events.

8. The method of claim 7 in which after the events are handled, the thread that is handling the events changes the bit array from the "no events and a thread is handling events" indication to the "no events and no thread is handling events" indication.

9. The method of claim 8 in which the thread that is handling events changes the bit array to the "no events and no thread is handling events" indication and terminates and wherein said thread ceases event handling.

10. The method of claim 8 in which upon condition that the thread that is handling events is unable to successfully change said indication, the thread repeatedly attempts said change until it succeeds in saving the indication that is stored in the bit array and changes the bit array to the "no events and a thread is handling events" indication.

11. The method of claim 6 in which, upon condition that the thread is unsuccessful in changing the bit array value, the thread makes a local copy of the bit array.

12. The method of claim 11 in which the thread adds events that it was requested to handle to the local indication in the bit array and employs the compare and swap function to attempt to replace an old indication in the bit array with a new local indication.

13. The method of claim 12 in which upon condition that the local bit array indication is replaced, the first thread is terminated, whereby the first thread has provided a mechanism to have the thread that is currently handling events handle subsequent events.

14. The method of claim 12 in which, upon condition that the local bit array indication is not replaced, the compare and swap instruction is employed to change the bit array from the "no events and no thread handling events" indication to the "no events and a thread is handling events" indication.

15. The method of claim 2 in which a list of events is updated by the compare and swap function to contain a list of events to be handled by the thread that is currently handling events.

16. The method of claim 15 in which the list of events includes an indication which signifies that no events are waiting to be handled and that no thread is currently handling events.

17. The method of claim 15 in which the list of events includes an indication that no events are waiting to be handled and that a thread is currently handling events.

18. The method of claim 16 in which the compare and swap instruction is employed to change the list of events from the "no events and no thread handling events" indication to the "no events and a thread is handling events" indication.

19. The method of claim 18 in which the list of events is changed to the "no events and a thread is handling events" indication so that the thread that makes that change handles subsequent events.

20. The method of claim 19 in which after the events are handled, the thread that is handling the events changes the list of events from the "no events and a thread is handling events" indication to the "no events and no thread is handling events" indication.

21. The method of claim 20 in which the thread that is handling events changes the list of events to the "no events and no thread is handling events" indication and wherein said thread ceases event handling.

22. The method of claim 20 in which, upon condition that the thread that is handling events is unable to successfully change said indication, the thread repeatedly attempts said change until it succeeds in saving the indication that is stored in the list of events and changes the list of events to the "no events and a thread is handling events" indication.

23. The method of claim 18 in which, upon condition that the thread is unsuccessful in changing the list of events, the thread makes a local copy of the list of events.

24. The method of claim 23 in which the thread adds events that it was requested to handle, to the local indication in the list of events and employs the compare and swap functionality to attempt to replace an old indication in the list of events with a new local indication.

25. The method of claim 24 in which, upon condition that the list of events is updated, the first thread is terminated, whereby the first thread has provided a mechanism to have the thread that is currently handling events handle subsequent events.

26. The method of claim 24 in which, upon condition that the list of events is not replaced, the compare and swap instruction is employed to change the list of events from the "no events and no thread handling events" indication to the "no events and a thread is handling events" indication.

27. A method for handling events in a multithreaded data processing system which comprises the step of insuring that only one thread gains control at a time so that a first thread, which has an event that needs to be handled at the time that a second thread is handling said event, passes the handling of the events from the first thread to the second thread, whereby the first thread does not need to wait for the second thread to finish and whereby no thread waits for a significant amount of time for another thread to finish.